REMARKS

Applicant is in receipt of the Office Action mailed November 18, 2004. Claims 7, 24, 27, and 30 have been cancelled. Claims 1, 8, 12-15, 17-23, 25, 26, 28, 29, and 31-33 have been amended. New claim 36 has been added. Thus, claims 1, 3-6, 8, and 12-23, 25, 26, 28, 29, and 31-36 remain pending in the case. Reconsideration of the present case is earnestly requested in light of the following remarks.

Objections to the Specification

The Specification was objected to due to omission of an application number in two paragraphs. These paragraphs have been amended to add the application number.

Applicant also found a similar omission in the paragraph beginning on page 49, line 4, which has also been amended accordingly.

The Abstract was amended to meet the 150-word limit, and to remove uses of the term "may". The Office Action indicated that the use of "may" in the entire Specification is objectionable, citing MPEP 608.01(b). However, Applicant notes that this portion of the MPEP is directed to the Abstract only. Applicant thus submits that the amended Abstract provided above addresses the objection.

Removal of the objections to the Specification is respectfully requested.

Section 112 Rejection

The Office Action rejected claims 1, 3-8, and 12-35 for failing to particularly point out and distinctly claimed the subject matter which Applicant regards as the invention. Applicant respectfully disagrees.

The Office Action asserts that it is unclear whether this graphical program is a placeholder/framework or a complete graphical program, and that the specification does not precisely described how to generate a complete graphical program, citing page 8 of the Background, which reads

> A state diagram may not explicitly specify the program instructions or functionality to be performed when each state is active, and thus it may not be possible to generate a

> complete graphical program implementing the functionality represented by the state diagram.

Applicant notes that the very next portion of the Background recites:

> In this instance, users may greatly benefit from the automatic generation of a graphical program "skeleton" that provides a framework for the various states and transitions (and relationships among these) of the state diagram. The desired source code for each state and the desired source code specifying the conditions for each transition can then be filled in (manually) to complete the graphical program.

Thus, while the Background states that "it would be desirable to provide a system and method for programmatically generating a graphical program based on a state diagram, with little or no user input required", the Background further states that: "users may greatly benefit from the automatic generation of a graphical program "skeleton" that provides a framework for the various states and transitions (and relationships among these) of the state diagram" (page 8, lines 21-23).

Furthermore, the description of Figure 5 on page 27, lines 25-27, states: "one embodiment of the present invention comprises a system and method for programmatically generating a graphical program (or portion of a graphical program) without requiring this type of user interaction."

In other words, the application clearly indicates that various embodiments of the invention include generation of a complete graphical program, and/or generation of a portion of a graphical program, i.e., a "framework" to which other graphical source code may be added, e.g., by the user. Applicant submits that in the art of text-based programming, the notion of a "stub program" is well-known, where the stub program includes an overall program structure or framework, but only function stubs for some or all of the functions used in the program, where the "stubbed out" functions are generally intended to be fleshed out with source code later, e.g., by the user.

The Office Action further asserts that the term "programmatically" is unclear, and that the Specification variously defines it as meaning "automatically" *and* as meaning "dynamically". Applicant respectfully submits that the Examiner has misread Applicant's use of the term. For example, the description of Figure 5 on page 29, lines 20-22, states: "In step 204, the GPG program may automatically, i.e., programmatically, generate a graphical program (or graphical program portion or graphical program framework) based on the received state diagram information." Note that "i.e." means "in other words".

Applicant notes that in the Background section, *manual* creation of a graphical program according to the prior art is described, referring specifically to U.S. Patent Nos. 4,901,221; 4,914,568; 5,291,587; 5,301,301; and 5,301,336; among others, to Kodosky et al., and as quoted above, states that "In contrast, it would be desirable to provide a system and method for programmatically generating a graphical program based on a state diagram, with little or no user input required." In other words, the programmatic generation of the graphical program is in contrast with the manual creation of the graphical program (e.g., by the user dragging and dropping selected graphical program nodes or icons onto the block diagram). Applicant further notes that the term "automatically" is clearly indicated as being a synonym for "programmatically", per the user of "i.e." in the above phrase "the GPG program may *automatically*, i.e., *programmatically*, generate a graphical program (or graphical program portion or graphical program framework) based on the received state diagram information".

Applicant submits that the Application is clear in its use of the terms "programmatically" and "automatically", and, as the Examiner states in the Office Action, that these terms refer to an action being performed by a program, as opposed to being performed manually, e.g., by the user. However, to clarify and emphasize this intended (and supported) meaning, the claims have been amended above to replace the term "programmatically" with "automatically".

As for the Examiner's assertion that the application asserts equivalence between "programmatically" and "dynamically", Applicant respectfully submits that this is not the case. For example, page 11, lines 24-26, read: "In one embodiment, the graphical program may be dynamically (programmatically) updated as the state diagram is being interactively constructed by the user"; and page 12, lines 2-5 read: "As the user performs various actions in the state diagram editor, such as adding or deleting states, adding or deleting transitions, changing the destination state of a transition, etc., the corresponding graphical program may be dynamically (programmatically) updated to reflect the change."

Note that the term "i.e." is *not* used in the parenthetical term, and thus the phrase "in other words" is *not* implied. Rather, the parenthetical use of "programmatically" here refers to the idea that while the user modifies the state diagram, the corresponding graphical program is dynamically updated *by a program*, i.e., *automatically*, as opposed to *manually*.. In other words, the term "dynamically" refers to the fact that the updates to the graphical program occur *as the state diagram is being modified by the user*, and the term "programmatically" refers to the fact that this dynamic updating is performed by a program, as opposed to a person performing the update *manually*.

Applicant notes that the program could have been updated dynamically, but not programmatically, e.g., by having a second user track the changes made to the state diagram in real time, and the second user manually modifying the program accordingly. This would be an example of dynamically updating the program where the updating is *not* performed programmatically. Thus, Applicant respectfully submits that the term "programmatically" is not indefinite as used in the present application.

The Office Action also asserts that the claims are indefinite because "it is unclear how the state diagram is received and how/where (any user interface?) it is created and sent to generate the corresponding graphical program". Applicant respectfully disagrees.

Creation of state diagram information is described on page 28, line 3 through page 29, line 3, which reads:

> In step 200, state diagram information may be created, e.g., in response to user input. For example, as described above with reference to Figure 2, the user may utilize a state diagram editor program that allows the user to create a state diagram. The state diagram editor may include a graphical user interface that displays the current state of the state diagram as the diagram is interactively constructed. In other embodiments, the state diagram information may be created in any of various other ways. A program having a graphical user interface that displays a state diagram corresponding to the state diagram information is not necessarily used. For example, the state

> diagram information may simply be created as a text file structured in such a way as to specify the states, state transitions, etc.
>
> As used herein, the term "state diagram information" comprises information that specifies at least: one or more states; and one or more state transitions, wherein each state transition specifies a transition from a first state to a second state (wherein the first and second states may be the same). In various embodiments, the state diagram information may also specify information such as: an initially active start state; one or more stop states; a priority ordering for transitions exiting from a given state; etc.
>
> The state diagram information preferably represents a state diagram (also called a "state flow diagram"), wherein the state diagram models or represents a desired algorithm or method. The state diagram information may also represent a sequential function chart (SFC). A sequential function chart is similar to a state diagram, with one exception being that multiple states can execute concurrently. The state diagram may represent desired operation of a software program or desired operation of a hardware device, or both. In one embodiment, the state diagram may represent a higher level representation of a desired algorithm than even a graphical program, and the present invention may be used to convert the state diagram into a graphical program or a graphical program framework.
>
> Another example of a state diagram is a test sequence, e.g., where the test sequence is represented as a plurality of states representing the respective tests in the sequence as well as the transitions between the steps. As used herein, a "test sequence" specifies a sequence of executable test modules as well as transitions or flow between the test modules, wherein the test sequence is used to test a unit under test.

Thus, the application clearly describes ways that the state diagram information can be created, e.g., via a state diagram editor program, and specifically mentions a graphical user interface usable by a user to do so. Moreover, page 29, lines 4-19 recites:

> In step 202, a graphical program generation (GPG) program may receive the state diagram information created in step 200. It is noted that the GPG program is not necessarily separate from a program used to create the state diagram information. For example,

> a single program or application may include functionality for both creating the state diagram information and programmatically generating the graphical program (and/or invoking a server program to cause the graphical program to be programmatically generated). In other embodiments, the GPG program may be a separate program.
>
> Thus, in various embodiments the GPG program may receive the state diagram information in any of various ways and from any of various types of sources. For example, the GPG program may be operable to read the state diagram information from a memory location in which the state diagram information was stored, or the state diagram information may be passed to the GPG program programmatically using standard programming techniques, or the GPG program may be operable to read the state diagram information from a file, etc. Also, in various embodiments, the received state diagram information may be structured or formatted in any way and may comprise various types of data, such as text data, binary data, XML data, etc.

Thus, the application describes various ways and means whereby the state diagram information may be created and received. Applicant notes that the particular manner in which the state diagram information is created and received is not limited to any specific way or means, and that this is why the independent claims do not include limitations restricting the manner in which the state diagram information is created and received. Applicant submits that this type of generality is not the same as the "indefiniteness" referred to in Section 112. In other words, the novelty and usefulness of the invention does not depend upon the particular manner in which the state diagram information is created and received. Applicant submits that the independent claims include features and limitations that are both novel and useful, and that render the claims definite per Section 112.

The Office Action further asserts that the claims are indefinite because " in line 4, it is unclear whether the state diagram specifies only a plurality of states without transitions between the states?[sic]". Applicant respectfully submits that claim 1 leaves this issue open, but certainly does not preclude the state diagram information specifying transitions between states. In fact, dependent claim 14 specifically provides this limitation. Applicant again submits that this type of generality is not what is meant by "indefiniteness" in Section 112.

The Office Action further asserts that the claims are indefinite because "in line 7, it is unclear how the graphical source code is programmatically generated based on the state diagram information".

Applicant respectfully submits that the present application includes numerous descriptions of techniques for programmatically or automatically generating the graphical source code based on the state diagram information. For example, regarding creation of the initial graphical program, page 34, lines 9-17, read:

> In one embodiment, the state diagram editor may interface with a GPG server program in order to request or direct the GPG server program to generate the initial graphical program. The GPG server program may provide an application programming interface (API) for programmatically generating graphical programs. In one embodiment, the GPG server program may be a graphical programming development environment application. For example, the LabVIEW graphical programming development environment application provides an API which client programs may use to programmatically generate or modify graphical programs.

and lines 22-29 read:

> In step 230, the state diagram editor program may receive user input specifying a change to the state diagram. Various types of possible changes are listed above (e.g., adding a state, adding a transition, etc.) In step 232, in response to the state diagram change performed in step 230, the state diagram editor may programmatically update the graphical program to correspond to the specified change, e.g., by calling an API of the GPG server program to perform the update. Thus, the API may enable not only the creation of a new graphical program, but may also allow modifications to an existing graphical program.

Applicant notes that Figures 8-18 provide examples of programmatically or automatically generated graphical source code.

Moreover, page 44, line 26 through page 49, line 15, describes at least one way in which the graphical source code may be programmatically or automatically generated based on the state diagram information, including one embodiment in which a client system makes API calls (text-based, and/or graphical) to a server system (based on the state diagram information) which then inserts the corresponding appropriate graphical source code into the graphical program. Applicant further notes that the section beginning on page 44, line 26 titled "<u>State Diagram Information Including States Which Specify Executable Code</u>" particularly describes graphical source code being associated with various states in the diagram, stating: "In one embodiment of the invention, the state diagram information may include information specifying executable code or source code, referred to below as program code, associated with one or more states." The application goes on to describe an embodiment where the user specifies the associations between states and source code, and states:

> "when the user associates program code with a state in a state diagram, the GPG program may operate to analyze the various states to which the respective state is connected and may use this information in "connecting" the graphical code created corresponding to this program code with the graphical code corresponding to the other states or with the graphical code created based on the program code associated with the other connecting states. Thus, the GPG program may traverse the state diagram and utilize the connections to "tie in" or connect the graphical program portion corresponding to the program code of a respective state with its neighboring or connecting states."

Applicant thus respectfully submits that the application does in fact provide descriptions of how the graphical source code may be programmatically or automatically generated based on the state diagram information.

The Office Action noted that claims 8 and 31 lack sufficient antecedent basis for "the specified one or more states". Applicant has amended claims 8 and 31 accordingly.

The Office Action also indicated that claim 12 is unclear regarding "for at least one state…". Applicant has amended claim 12 accordingly.

Regarding claim 23, the Office Action indicated that "first functionality" is unclear. Applicant notes that the term "first" is simply used as a label, e.g., to distinguish from any subsequently introduced "functionalities". Applicant also notes that, per claim 23, the "first functionality" specified by the state diagram is implemented by the generated graphical program. In other words, the claim makes clear that the generated graphical program correlates functionally with the state diagram. Applicant respectfully submits that such use of "first functionality" is not indefinite, but rather is simply an expression of the idea that the state diagram and the graphical program may respectively specify and implement any functionality desired.

Thus, for at least the reasons provided above, Applicant respectfully submits that the Section 112 rejection of claims 1, 3-8, and 12-35 is rendered moot, and respectfully requests removal of the Section 112 rejection of these claims.

Section 102 Rejections

Claims 1-34 were rejected under 35 U.S.C. 102(b) as being anticipated by MathWorks ("Stateflow for State Diagram Modeling User's Guide", version 4, 1997-2001, henceforth, "MathWorks"). Applicant respectfully disagrees.

As the Examiner is certainly aware, anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim. *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The identical invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). Moreover, an 'anticipating' reference must describe all of the elements and limitations of the claim in a single reference, and enable one of skill in the field of the invention to make and use the claimed invention. *Bristol-Myers Squibb Co. v. Ben Venue Labs., Inc.*, 246 F.3d 1368, 1378-79 (Fed. Cir. 2001); *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226 (Fed. Cir. 1989)." *In re Merck & Co., Inc. v. Teva Pharm. USA, Inc.*, 347 F.3d 1367, 1372 (Fed. Cir. 2003).

Amended claim 1 recites:

1. A computer-implemented method for automatically generating a graphical program based on a state diagram, comprising:

receiving state diagram information, wherein the state diagram information represents the state diagram and specifies a plurality of states;

automatically generating the graphical program in response to the state diagram information, wherein said automatically generating comprises automatically generating graphical source code corresponding to the plurality of states, wherein the graphical source code comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein the graphical program is executable by a computer, and wherein said automatically generating the graphical program creates the graphical program without any user input specifying the graphical program during said creating.

The Office Action asserts that MathWorks discloses all the features and limitations of claim 1. For example, the Office Action asserts that MathWorks discloses "programmatically generating the graphical program in response to the state diagram information, wherein said programmatically generating comprises programmatically generating graphical source code corresponding to the plurality of states, wherein the graphical source code comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program", citing the State Flow manual section called "Creating a Simulink Model" which reads " these steps described how to create a simulated model with a Stateflow block, label the Stateflow block, and save the model:", and "A Simulink model can consist of combinations of Simulink blocks, toolbox blocks, and Stateflow blocks," (page 2-4; and the figure in 2-7).

Applicant respectfully disagrees.

Applicant respectfully submits that MathWorks actually discloses including a Stateflow block in a Simulink model, where the Stateflow block and the Simulink model operate in conjunction. As the Office Action itself describes, "The Simulink model and Stateflow machine work seamlessly together. Running a simulation automatically executes both the Simulink and State flow portions of the model" (page 2-4).

The cited portions of the Stateflow manual clearly describe manual inclusion of a Stateflow block in a Simulink model diagram, and state in part: "you can either start with the default in the model or copy the untitled Stateflow block into any Simulink model to include a Stateflow diagram in an existing Simulink model". In other words, the Stateflow block is used as an element in the Simulink model, and is *not* used as a basis for programmatically or automatically generating graphical program source code. Nowhere does MathWorks teach or describe programmatically or automatically generating graphical program source code based on received state diagram information.

Thus, Applicant submits that MathWorks fails to teach or suggest all the features and limitations of claim 1, and so, for at least the reasons provided above, Applicant respectfully submits that claim 1 and those claims dependent therefrom are patentably distinct and non-obvious over MathWorks, and are thus allowable.

Independent claims 23, 25, 26, 29, and 32 include similar limitations as claim 1, and so the above arguments apply with equal force to these claims. Additionally, claim 32 includes the limitations that "wherein a first one or more nodes comprise graphical source code executable to implement first functionality corresponding to a first one or more states, and wherein a second one or more nodes are user-configurable to implement second functionality of a corresponding second one or more states". The second one or more nodes correspond to the "framework" embodiments described above, in which basic structure of the graphical program is programmatically or automatically generated, but source code for these nodes is provided or configured by the user. Nowhere does MathWorks teach or suggest this feature.

Thus, for at least the reasons provided above, Applicant respectfully submits that claims 23, 25, 26, 29, and 32, and those claims respectively dependent therefrom, are patentably distinct and non-obvious over MathWorks, and are thus allowable.

Removal of the 102 rejection of claims 1-34 is respectfully requested.

Section 103 Rejections

Claim 35 was rejected under 35 U.S.C. 103(a) as being unpatentable over MathWorks ("Stateflow for State Diagram Modeling User's Guide", version 4, 1997-2001, henceforth, "MathWorks") in view of Kodosky et al (US 5,732,277, "Kodosky"). Applicant respectfully disagrees.

To establish a prima facie obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. In re Royka, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. Obviousness cannot be established by combining or modifying the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion or incentive to do so. In re Bond, 910 F. 2d 81, 834, 15 USPQ2d 1566, 1568 (Fed. Cir. 1990).

Moreover, as held by the U.S. Court of Appeals for the Federal Circuit in Ecolochem Inc. v. Southern California Edison Co., an obviousness claim that lacks evidence of a suggestion or motivation for one of skill in the art to combine prior art references to produce the claimed invention is defective as hindsight analysis.

In addition, the showing of a suggestion, teaching, or motivation to combine prior teachings "must be clear and particular . . .. Broad conclusory statements regarding the teaching of multiple references, standing alone, are not 'evidence'." *In re Dembiczak*, 175 F.3d 994, 50 USPQ2d 1614 (Fed. Cir. 1999). The art must fairly teach or suggest to one to make the specific combination as claimed. That one achieves an improved result by making such a combination is no more than hindsight without an initial suggestion to make the combination.

Finally, as stated in the MPEP §2143.01, "The mere fact that references <u>can</u> be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990). . .

Claim 35, amended independent claim 32, and amended intervening claim 33 recite:

32. A computer-implemented method for automatically generating a graphical program based on state diagram information, comprising:

receiving the state diagram information, wherein the state diagram information specifies a plurality of states and transitions between the states;

automatically generating the graphical program in response to the state diagram information, wherein the graphical program comprises a plurality of interconnected nodes which visually indicate functionality of the graphical program, wherein a first one or more nodes comprise graphical source code executable to implement first functionality corresponding to a first one or more states, wherein a second one or more nodes are user-configurable to implement second functionality of a corresponding second one or more states, and wherein said automatically generating the graphical program creates the graphical program without any user input specifying the graphical program during said creating.

33. The method of claim 32,

wherein the automatically generated graphical program includes placeholder graphical source code for each of the second one or more states.

35. The method of claim 33,

wherein the placeholder graphical source code for each state comprises a case in a graphical case structure.

Applicant respectfully submits that neither MathWorks nor Kodosky provides a motivation to combine. For example, nowhere does MathWorks teach or suggest, or indicate the desirability of, *a second one or more nodes are user-configurable to implement second functionality of a corresponding second one or more states*, *wherein the automatically generated graphical program includes placeholder graphical source code for each of the second one or more states*, and *wherein the placeholder graphical source code for each state comprises a case in a graphical case structure.* Nor does Kodosky teach or suggest, or indicate the desirability of, these features. Nothing disclosed in either of the cited references provides or suggests a motivation to combine the references. Thus, Applicant submits that the attempted combination of MathWorks and Kodosky is improper.

Additionally, Applicant submits that even were MathWorks and Kodosky properly combinable, which Applicant argues they are not, the resulting combination would still not teach Applicant's invention as represented in claim 35. For example, neither reference discloses or even hints at the programmatic or automatic generation of graphical program source code based on received state diagram information, and more specifically, neither reference teaches or describes: *wherein a second one or more nodes are user-configurable to implement second functionality of a corresponding second one or more states, wherein the automatically generated graphical program includes placeholder graphical source code for each of the second one or more states*, and *wherein the placeholder graphical source code for each state comprises a case in a graphical case structure.*

Furthermore, Applicant notes that if an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Applicant submits that independent claim 32 has been shown above to be patentably distinct and non-obvious, and so dependent claim 35 is similarly patentably distinct and non-obvious.

Thus, for at least the reasons provided above, Applicant respectfully submits that MathWorks and Kodosky, taken singly or in combination, fail to teach or suggest all the features and limitations of claim 35, and so claim 35 and any claims dependent therefrom are patentably distinct and non-obvious over MathWorks and Kodosky.

Removal of the 103 rejection of claim 35 is earnestly requested.

New Claims

Applicant submits new claims 36 – 54 to more fully and completely claim Applicant's invention. Applicant submits that these claims are supported by the original specification and are allowable for at least the reasons given above.

Applicant also asserts that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the independent claims have been shown to be patentably distinct, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicant submits the application is in condition for allowance, and an early notice to that effect is requested.

If any extensions of time (under 37 C.F.R. § 1.136) are necessary to prevent the above referenced application(s) from becoming abandoned, Applicant(s) hereby petition for such extensions. If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert & Goetzel PC Deposit Account No. 50-1505/5150-45900/JCH.

Also enclosed herewith are the following items:

☒ Return Receipt Postcard

Respectfully submitted,

[illegible]

Jeffrey C. Hood
Reg. No. 35,198
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert & Goetzel PC
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8800
Date: 1/13/2005 JCH/MSW